UNIVERSIDAD NACIONAL DEL LITORAL

Facultad de Ingeniería y Ciencias Hídricas



Ingeniería en Inteligencia Artificial

Introducción a los Sistemas Cibernéticos

"Proyecto de Compostaje"

Equipo Docente: Juan Carrique, Eugenio Borzone, Bruno Zorzet, Tomás Molas.

Alumnos: Ciola, Avril Victoria -Durizotto, Nadia - González, Milagros - Luna, Nicolas - Maidana, María Micaela - Suarez, Irene - Sosa, Valentín

20/11/2024

Resumen:

En el proceso de compostaje, es importante tener un buen control del estado y las condiciones en las que se desarrolla. Para ello, en este trabajo se desarrolla un gemelo digital de un sensor de temperatura y diversos actuadores controlados mediante un microcontrolador. Este sistema no solo permite medir la temperatura en tiempo real, sino también identifica la etapa del compostaje en la que se encuentra el proceso.

El sistema utiliza un microcontrolador Arduino junto con un sensor de temperatura DHT22, un sensor de movimiento PIR HC-SR501, un reloj en tiempo real DS1307, y un display LCD compatible con HD44780. En este trabajo se analizan las características de cada componente y se produce un diagrama de estado que define las transiciones entre las etapas del proceso. Además, se realiza el gemelo digital en la plataforma “Wokwi.com”, donde se implementan los algoritmos principales para cada etapa del compostaje.

De esta forma, en el presente proyecto se sostiene que el sistema que se efectúa es eficiente, de bajo costo y fácil de implementar gracias al uso de sensores de alta precisión y estrategias de optimización de energía que posibilitan realizar un monitoreo continuo, contribuyendo al manejo sostenible del compostaje.

Objetivos:

- Desarrollar un sistema automatizado de monitoreo.
- Medir y registrar la temperatura del compostaje.
- Detectar cambios en el estado.
- Mostrar de manera clara las lecturas de temperatura y estad

Descripción del Sistema:

El uso de distintos actuadores y sensores es fundamental para posibilitar la creación de un sistema que realice la medición y la respectiva presentación de datos al cliente. Para esto hemos buscado los candidatos compatibles para el mismo y diferentes proveedores dentro del país.

Proveedores:

En Argentina existen varias tiendas tanto físicas como sitios web especializadas en la compra de sensores como actuadores, aquí hay algunas recomendaciones:

- **SEMAK SA:** SEMAK SA es una empresa nacional dedicada a la distribución de componentes electrónicos, equipos y accesorios para proyectos y desarrollos en diversas áreas tecnológicas. trabaja con marcas reconocidas como Arduino, Raspberry Pi, Vishay, Infineon, y otras enfocadas en tecnologías innovadoras. como servicios adicionales tiene asesoramiento técnico en la elección de productos. A continuación, un resumen de lo que ofrecen:
    1. **Componentes electrónicos:** Incluye semiconductores, optoelectrónica, pasivos y electromecánicos/magnéticos.
    2. **Sensores y módulos:** Para robótica, automatización e IoT.
    3. Placas de desarrollo: Dispositivos como Arduino, Raspberry Pi y otros sistemas embebidos.
    4. **Sistemas inalámbricos e IoT:** Productos para comunicación WiFi, LoRaWAN, GSM, Bluetooth y Zigbee.
    5. **Robótica y prototipos:** Incluye motores, servos, protoboards y cables para proyectos avanzados.
    6. **Accesorios y herramientas:** Como estaños, fuentes de alimentación, coolers y otros elementos esenciales para laboratorios electrónicos

    Sitio oficial: https://www.semak.com.ar/

- **ELECTRONICA ELEMON:** Con sucursales en Buenos Aires y Córdoba, esta tienda tiene un catálogo completo de sensores y otros componentes electrónicos para aplicaciones

industriales y proyectos de robótica. Se puede comprar tanto en persona como online.. Entre sus productos principales que ofrecen son:

1. **Sensores:** Disponen de sensores de temperatura, proximidad, humedad, luz, movimiento, y más, para aplicaciones en proyectos electrónicos o industriales.
2. **componentes electrónicos:** podemos hallar componentes pasivos (resistencias, capacitores, inductores), activos (Transistores, diodos, y circuitos integrados)
3. **Instrumentos de medición y herramientas:** Multímetros, fuentes de alimentación, y soldadores.
4. **Accesorios eléctricos e industriales:** Conectores, cables, fuentes, y más.

Sitio oficial: https://www.elemon.com.ar

- **GM ELECTRÓNICA:** Su catálogo abarca una amplia variedad de componentes electrónicos y herramientas. entre ellas están:
    1. **Sensores:** podemos encontrar de proximidad (inductivos, capacitivos y fotoeléctricos), presión, temperatura y flujo.
    2. **Instrumentos de medición**: Multímetros, amperímetros, osciloscopios, y termómetros.
    3. **Componentes electrónicos**: Semiconductores, circuitos integrados, y conectores. Cables y otros accesorios eléctricos.
    4. **Iluminación y señalización**: Tiras de LED y dispositivos para entornos industriales.

    Sitio oficial: https://gmelectronica.com.ar/sensores-inductivos-capacitivos/

- **TODOMICRO:** especializada en la venta de componentes electrónicos, sensores, actuadores y piezas electromecánicas. Ofrece una amplia variedad de productos que abarcan desde componentes básicos hasta soluciones avanzadas para proyectos complejos. Está enfocada tanto para aficionados como profesionales en el área y sus tiendas físicas se encuentran en CABA.
    1. **Sensores**: Sensores de temperatura y humedad, de proximidad, movimiento y ultrasonido, entre otros.
    2. **Actuadores**: Motores DC, servomotores y motores paso a paso. Relés y módulos de control de potencia y Bombas de agua y actuadores lineales.

3. **Piezas electromecánicas**: Conectores, cables, y módulos adaptadores. Interruptores, botones, y displays LCD/LED. Tarjetas controladoras como Arduino, ESP8266, y Raspberry Pi.
4. **Otros componentes**: Resistencias, capacitores, transistores y circuitos integrados. Placas de desarrollo y protoboards.Fuentes de poder, cargadores y baterías.

Sensores y actuadores:

A continuación, se encuentran los distintos sensores ya actuadores escogidos y sus especificaciones técnicas.

- **DHT22, sensor de temperatura y de humedad:** Mide la humedad entre 0% y 100% y la temperatura entre -40ºC y 80ºC. En este caso nos enfocaremos exclusivamente en la medición de temperatura.

Modelo	DHT22
Fuente de alimentación	3.3-6V DC
Señal de salida	Señal digital a través de un bus único
Rango operativo (Humedad)	0-100% RH
Rango operativo (Temperatura)	-40 ~ 80°C
Precisión (Humedad)	±2% RH (máx. ±5% RH)
Precisión (Temperatura)	±0.5°C
Resolución (Humedad)	0,1% RH
Resolución (Temperatura)	0.1°C
Estabilidad a largo plazo (Zero Drift)	±0.5% RH/año
Periodo de muestreo	Promedio: 2s

Consumo típico: 1.5 mA para 5V.

Consumo promedio diario: 1.5mA*5V*24h=180mWh.

EJEMPLO DE APLICACIÓN Arduino:

```
#include "DHT.h"

#define DHTPIN 2
#define DHTTYPE DHT22

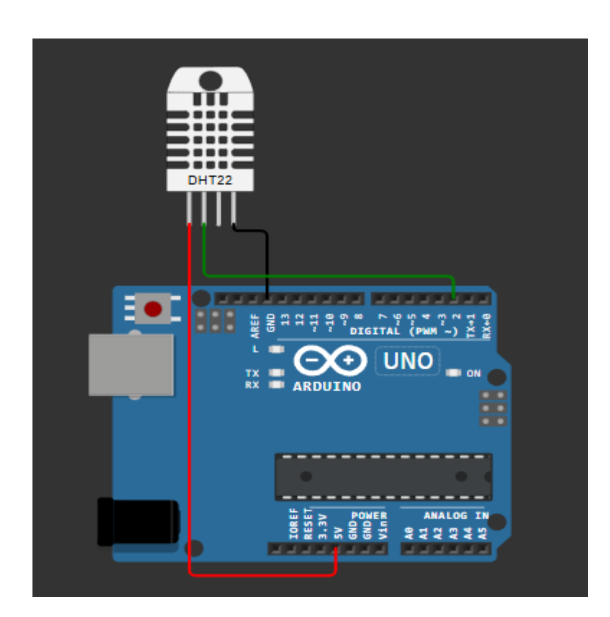
DHT dht(DHTPIN, DHTTYPE);

void setup() {
  Serial.begin(115200);
  dht.begin();
}

void loop() {
  float temperatura = dht.readTemperature();

  Serial.print(F("%  Temperature: "));
  Serial.print(temperatura);
  Serial.println(F("ºC "));

  delay(2000);
}
```



- **HC-SR501 sensor PIR (infrarrojo pasivo):** Detectan la radiación IR: es una forma de radiación electromagnética que no puede ser vista por el ojo humano, pero sí se puede sentir como calor que emiten o reflejan los objetos.

Modelo	HC-SR501
Fuente de alimentación	5V DC (generalmente alimentado por 5V)
Nivel de Salida	Alto: 3.3 V / Bajo: 0V
Tiempo de retardo	2.5 segundos (predeterminado) - ajustable de 0.xx a varios segundos
Ángulo del Sensor	<110° (ángulo del cono de detección)
Temperatura de Operación	-15°C a +70°C
Salida	HIGH/LOW

- Consumo en reposo: 50uA (operando a 5V).
- Consumo promedio (con una estimación de tiempo activo de una hora diaria):
  - En reposo:0.05mA*5V*1h=1,15mWh.
  - Activo:1mA*5V*1h=5mWh.
  - I promedio= (I reposo x T reposo + I activo *T activo)/T total.
  - I promedio = (1,15mWh x 23 + 5mWh *1)/24h.
  - Consumo promedio total por hora: 1,31 mAh (aproximación).

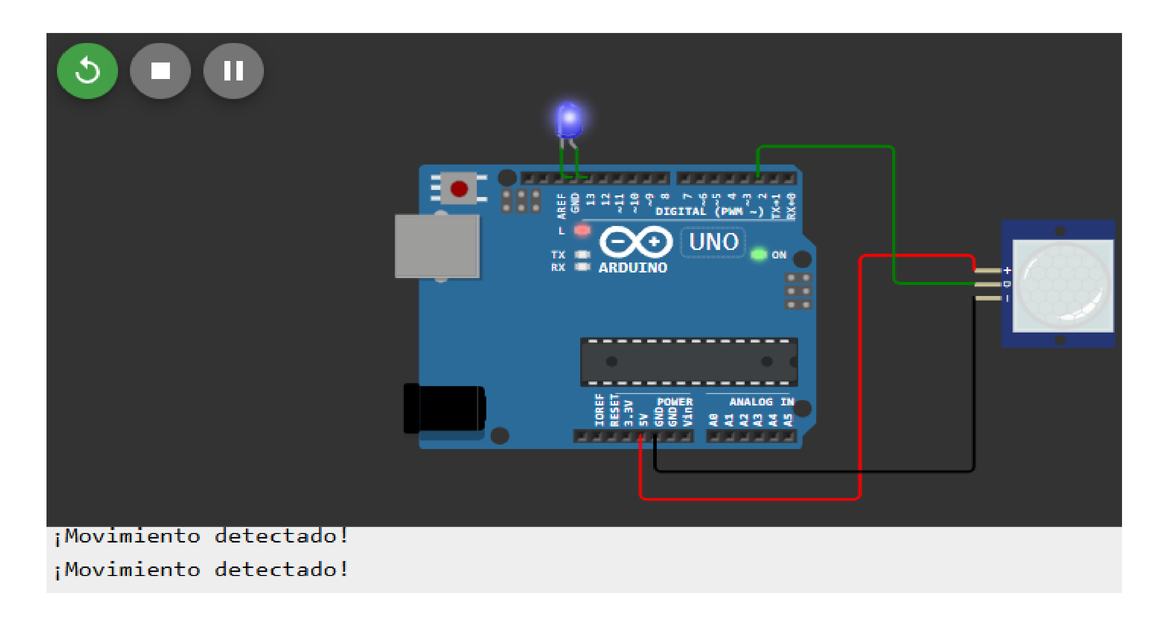
EJEMPLO DE APLICACIÓN Arduino:

```
const int pirPin = 2;   // Pin digital conectado al sensor PIR
const int ledPin = 13;  // Pin digital conectado al LED

void setup() {
  pinMode(pirPin, INPUT); // Configura el pin del PIR como entrada
  pinMode(ledPin, OUTPUT); // Configura el pin del LED como salida
  Serial.begin(9600); // Inicializa la comunicación serial
}

void loop() {
  int pirValue = digitalRead(pirPin); // Lee el valor del sensor PIR

  if (pirValue == HIGH) { // Si se detecta movimiento
    digitalWrite(ledPin, HIGH); // Enciende el LED
    Serial.println("¡Movimiento detectado!");
    delay(1000);
  } else {
    digitalWrite(ledPin, LOW); // Apaga el LED
  }
}
```



- **RTC DS1307 reloj en tiempo real:** guardar y recibir la hora y fecha actual.

Modelo	RTC DS1307
Voltaje de operación	2.0V - 5.5V
Temperatura de operación	-40°C a +85°C
Precisión	Mejor que ±2 minutos/mes a 25°C
Compensación de año bisiesto	Sí (hasta el año 2100)
Tipo de encapsulado	DS1307: 8 pines DIP o SOIC
Memoria no volátil	56 bytes de RAM no volátil para almacenamiento de datos
Interfaz de comunicación	Interfaz serie I2C

Consumo:

Máxima corriente de consumo: 1.5 mA.

Consumo para 5v: Potencia=5V x 1.5mA=7.5mW (consumo máximo).

Si el reloj está en constante funcionamiento el consumo total diario es de:

Consumo diario: 7.5mW×24h==180mWh/día.

EJEMPLO DE APLICACIÓN Arduino:

```
#include "RTClib.h"

RTC_DS1307 rtc;

char dias_semana[7][12] = {"Domingo", "Lunes", "Martes", "Miércoles",
"Jueves", "viernes", "Sabado"};

void setup () {

  Serial.begin(6900);

  if (! rtc.begin()) {

    Serial.println("No se encontró RTC");

    Serial.flush();

    abort();

  }



  if (! rtc.isrunning()) {

    Serial.println("iniciando el tiempo");

    rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));

  }

}

void loop () {

    DateTime now = rtc.now();

    Serial.print(now.year(), DEC);

    Serial.print('/');

    Serial.print(now.month(), DEC);
```

```
    Serial.print('/');

    Serial.print(now.day(), DEC);

    Serial.print(" (");

    Serial.print(dias_semana[now.dayOfTheWeek()]);

    Serial.print(") ");

    Serial.print(now.hour(), DEC);

    Serial.print(':');

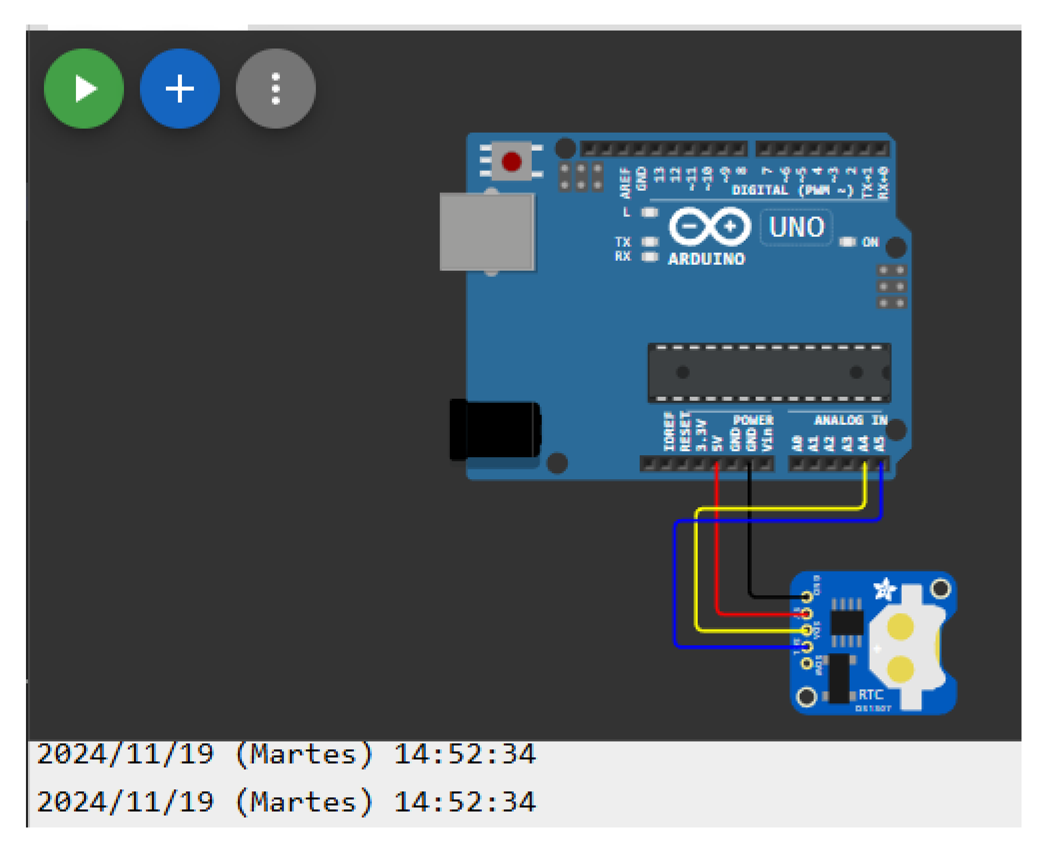
    Serial.print(now.minute(), DEC);

    Serial.print(':');

    Serial.print(now.second(), DEC);

    Serial.println();

}
```



VCC (RTC) → 5V (Arduino): Este pin del RTC se conecta al pin de **5V** del Arduino para alimentarlo. El DS1307 funciona con un voltaje de operación de 5V.

GND (RTC) → GND (Arduino): El pin de **GND** del RTC se conecta al pin **GND** del Arduino, completando el circuito eléctrico.

SDA (RTC) → A4 (Arduino): El pin **SDA** del RTC (Serial Data) se conecta al pin **A4** del Arduino Uno. Este pin se utiliza para transmitir y recibir datos en el bus I2C.

SCL (RTC) → A5 (Arduino): El pin **SCL** del RTC (Serial Clock) se conecta al pin **A5** del Arduino Uno. Este pin proporciona la señal de reloj necesaria para sincronizar la comunicación en el bus I2C.

- **Push Button (Botón Pulsador):**

Modelo	Tact Switch 6x6x5 mm
Voltaje de operación	5V
Corriente máxima soportada	0.5 A
Temperatura de operación	-20°C a 70°C
Resistencia de aislamiento	≥ 100 MΩ
Resistencia de contacto	≤ 100 mΩ
Durabilidad eléctrica	≥ 100,000 ciclos
Durabilidad mecánica	≥ 1,000,000 ciclos

Consumo durante la pulsación:

P=VxI=5V x 0.5A=2.5W

E=Pxt= 2.5Wx(5/3600 h)=3.47mWh.

Consumo en reposo: nulo.

El consumo promedio diario (una pulsación dividida 7 días): 3.47mWh/7=0.50mWh

EJEMPLO DE APLICACIÓN Arduino:

```
int boton = 4;

void setup() {

  pinMode(boton, INPUT); // Configura el pin como entrada

  Serial.begin(9600);    // Inicia la comunicación serial

}

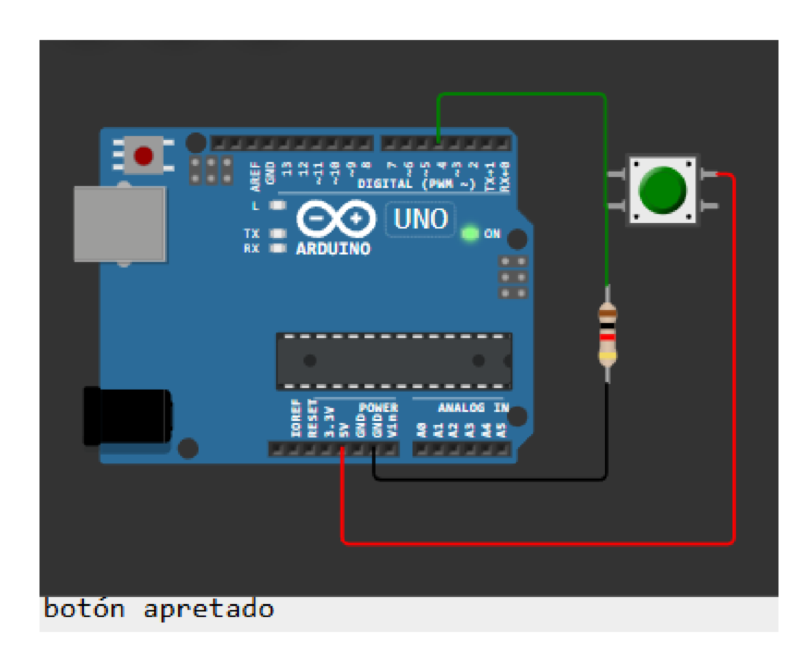
void loop() {

  if (digitalRead(boton) == HIGH) { // Detecta si el botón está presionado

    Serial.println("botón apretado");

  }

}
```



Modelo	HD44780U dot-matrix liquid crystal display
Voltaje de operación	Soporta de 2.7V a 5.5V
Rango de voltaje de operación del LCD	3.0V a 11V
Interfaz de bus MPU	Alta velocidad, hasta 2 MHz (cuando VCC = 5V)
Memoria RAM para pantalla	80 × 8 bits (máximo 80 caracteres)
RAM generadora de caracteres	64 × 8 bits
Controlador de pantalla	16 comunes × 40 segmentos
Funciones de instrucciones	Borrar pantalla, volver al inicio del cursor, encender/apagar pantalla, encender/apagar cursor, parpadeo de caracteres, desplazamiento del cursor, desplazamiento de la pantalla
Circuito de reinicio automático	Inicializa el controlador/driver después de encender
Bajo consumo de energía	Sí
ROM generador de caracteres	9,920 bits (240 fuentes de caracteres)

- **Pantalla LCD (Liquid Crystal Display) Controlador compatible con HD44780**

Voltaje de Alimentación: 5V.

Corriente de Consumo Máxima: 25 mA (0.025 A).

Uso de la pantalla: Se asume que está mostrando contenido continuamente.

Consumo máximo: P=V X I = 5V X 25mA=125W.

Consumo máximo diario: 0.125 W x 24h=3Wh/día.

Consumo realista: Si suponemos que la corriente no funcionará al máximo durante todo el día, el consumo se reduce. Con un promedio de consumo de 15mA(valor aproximado si reducimos el brillo y la retroiluminación).

Consumo promedio realista: 5v x 15 mA=75mA.

Consumo diario promedio: 75mAx24h=1.8Wh/día.

EJEMPLO DE APLICACIÓN Arduino:

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C Prueba(0x27, 20, 4);

void setup()
{
  Prueba.init();
  Prueba.backlight();
  Prueba.setCursor(0, 0);
  Prueba.print("    Hola Mundo :)!");
  Prueba.setCursor(0, 1);
  Prueba.print("    I2C LCD 20x4");
}

void loop()
{
}
```

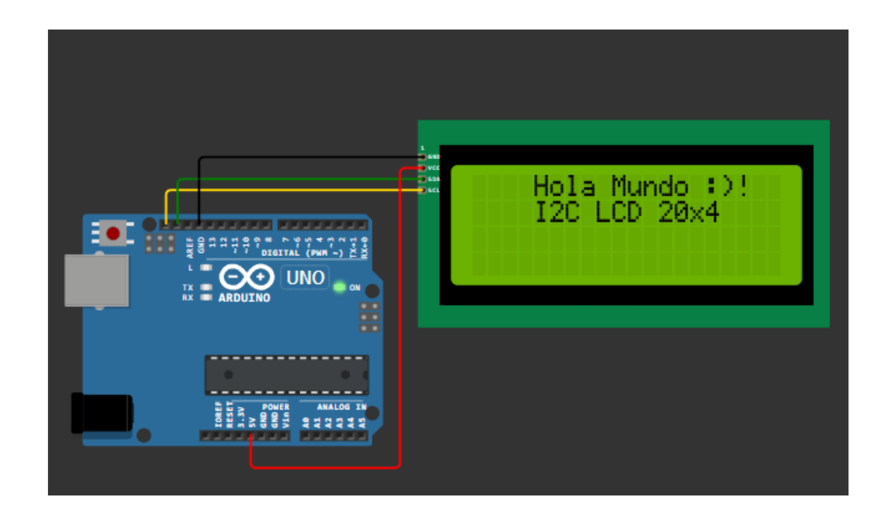


Tabla comparativa de Sensor de Temperatura:

Nombre	DHT22	DHT11	DS18B20	LM35
Alcance	-40 ~ 80°C	-0 ~ 50°C	-55 ~ 125°C	-55 ~ 150°C
Ventajas	Rango de temperatura	Económico	Resistente al agua	Alta precisión, económico
Desventajas	Menor velocidad de respuesta	Rango muy limitado	Requiere configuración extra.	Sensible a interferencias y requiere ADC
Motivo de elección/no elección	Alta precisión, rango adecuado y salida digital.	El compost alcanza temperaturas mayores a los	Su rango al ser más amplio que el requerido trae complejidad innecesaria, añadida a la de su protocolo de funcionamiento.	Al ser un sensor analógico requiere de procesamiento adicional además de ser propenso a interferencias.

Tabla comparativa de sensor infrarrojo:

característica	HC-SR501	TCRT5000	GP2Y0A21YK
función	Detección de movimiento por calor	Detección de objetos y proximidad	Medición de distancia por reflexión infrarroja
rango efectivo	3-7 metros (ajustable)	0.2-15 mm	10-80 cm
precio	Bajo (~2-4 USD)	Muy bajo (~1	Medio (~10 USD)

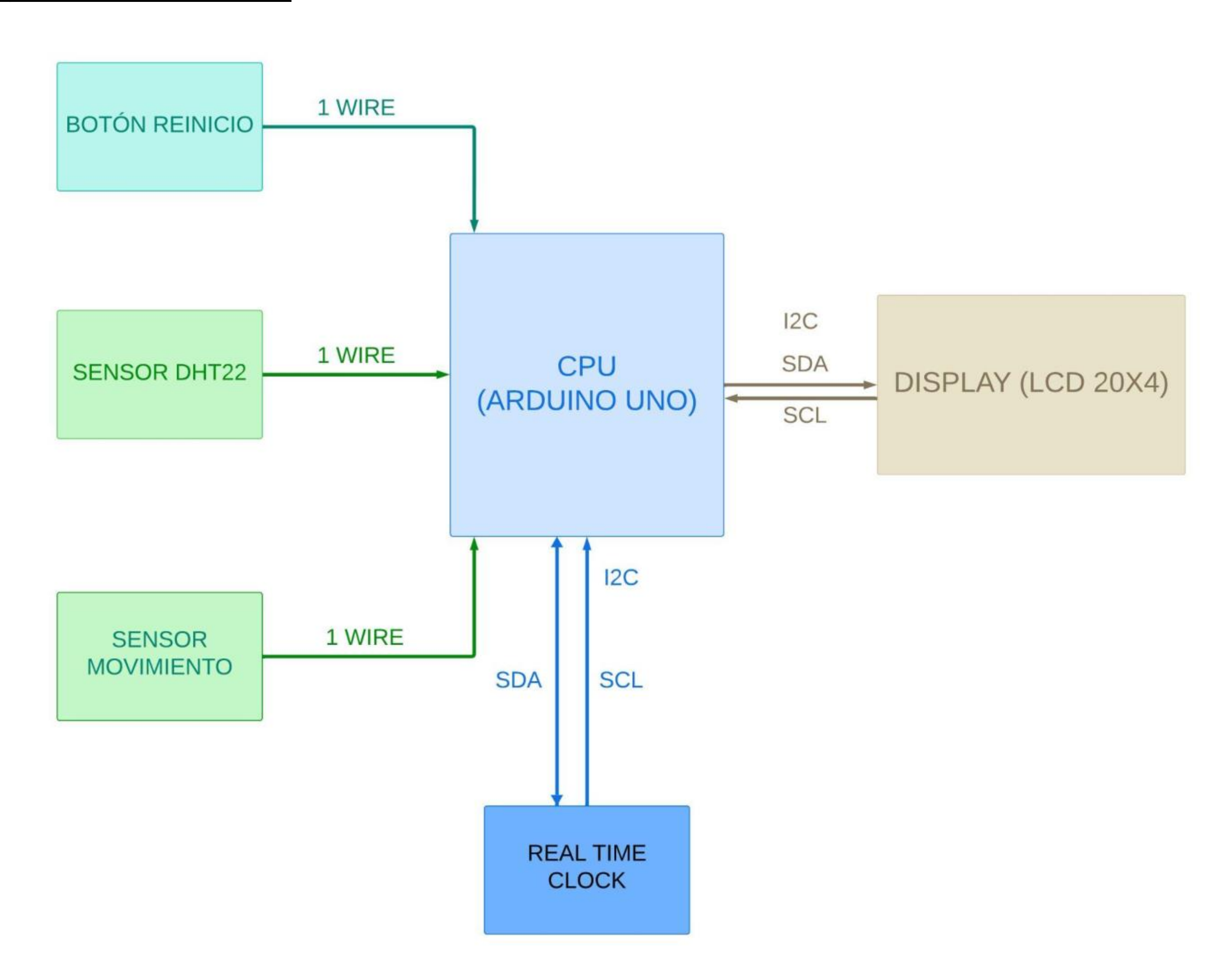
		USD)	
ventajas	Económico, largo alcance	Compacto, preciso en proximidad	Preciso para objetos dentro del rango
limitaciones	No mide distancia o temperatura	Rango limitado a pocos mm	No detecta bien objetos oscuros o transparentes

Tabla comparativa del controlador:

característica	HD44780U	ST7066U	KS0066U
Fabricante original	Hitachi (original y más conocido)	Sitronix (compatible con HD44780U)	Samsung (compatible con HD44780U)
compatibilidad	Estándar de referencia para LCD alfanuméricos	Totalmente compatible con HD44780U	Totalmente compatible con HD44780U
consumo de energía	Moderado	Más eficiente (menor consumo)	Similar al HD44780U
optimización interna	Diseño clásico	Mejor gestión de energía y señales	Puede requerir ajustes menores en tiempo de señales
disponibilidad	Muy amplia (base de módulos LCD comunes)	Común pero menos disponible que HD44780U	Menor disponibilidad que HD44780U y ST7066U
documentación	Muy amplia y detallada	Menos detallada pero suficiente	Documentación básica
velocidad de	Estándar	Ligeramente más	Similar al HD44780U

operación		rápida	

Diagrama de bloques:



Dentro de los DataSheets de cada sensor y actuador podemos encontrar la cantidad de líneas y protocolos que utilizan.

Botón Reinicio:

- Líneas: 1 línea (conexión simple a un pin de entrada digital del Arduino).

- Protocolo: No requiere protocolo de comunicación (envía una señal digital HIGH o LOW).
- No requiere alimentación externa.

Sensor DHT22: (Temperatura y Humedad):

- Líneas: 1 línea de datos.
- Protocolo: Comunicación single-wire protocol, utiliza un protocolo propietario para enviar datos de temperatura y humedad al Arduino.
- Alimentación 3.3V a 5V.

Sensor de Movimiento (PIR):

- Líneas: 1 línea de salida digital.
- Protocolo: No utiliza un protocolo de comunicación formal (envía una señal digital HIGH o LOW).
- Alimentación 5V a 12V.

Real Time Clock (DS1307):

- Líneas: 2 líneas (SDA y SCL).
- Protocolo: I2C. Este protocolo de comunicación serial permite la sincronización de datos con el Arduino usando dos líneas compartidas (SDA y SCL).

  Línea de reloj (SCL): Transporta pulsos de onda cuadrada que sincroniza la transferencia de datos

  Línea de datos (SDA): Transporta los datos entre los dispositivos.

- Alimentación 5V.

Display LCD 20x4 I2C:

- Líneas: 2 líneas (SDA y SCL)
- Protocolo: I2C. Se comunica con el Arduino a través del mismo bus I2C compartido por el RTC, utilizando direcciones I2C distintas para cada dispositivo.
- Alimentación 5V.

Variables usadas en el código:

1) **DHTPIN** (constante)

   Descripción: Define el pin digital donde está conectado el sensor de temperatura y humedad (DHT).

   Valor: 2 (pin digital).

2) **DHTTYPE** (constante)

   Descripción: Especifica el tipo de sensor DHT utilizado. En este caso, el DHT22.

3) **pirPin** (constante)

   Descripción: Define el pin digital donde está conectado el sensor de movimiento PIR.

   Valor: 4.

4) **buttonPin** (constante)

   Descripción: Define el pin digital donde está conectado el botón para resetear el sistema.

   Valor: 5.

5) **dht** (objeto)

   Descripción: Instancia del sensor de temperatura y humedad (DHT) que se inicializa con el pin y el tipo de sensor.

6) **contador_p** (variable int)

   Descripción: Lleva el conteo de las veces que se "airea" el compost, detectado por el sensor de movimiento.

7) **State** (enumeración)

   Descripción: Define los estados posibles del compostador: vacío, lleno, mesófila, termófila y finalizado.

8) **state** (variable de tipo State)

Descripción: Almacena el estado actual del compostador. Inicialmente se establece en “STATE_VACIO”.

9) **lcd** (objeto)

Descripción: Instancia del objeto para manejar el LCD (Liquid Crystal Display). Se inicializa con la dirección I2C y el tamaño del display (20x4).

10) **rtc** (objeto)

Descripción: Instancia para interactuar con el RTC (reloj de tiempo real) DS1307.

11) **startTime** (objeto DateTime)

Descripción: Almacena el momento en que se inicia un estado o se realiza un cambio de estado.

12) **temperature** (variable float)

Descripción: Guarda la temperatura actual leída del sensor DHT.

13) **currentTime** (objeto DateTime)

Descripción: Almacena la hora y fecha actuales proporcionadas por el RTC.

Comunicación/Filtros para la señal:

Nombre	**Conversión**	**Amplificación**	**Filtro anti-aliasing**
HD44780U (pantalla)	Compatible	No	No
Tact Switch (botón)	Compatible	No	No
RTC DS1307 (reloj)	Compatible	No	No
HC-SR501 (puerta)	Compatible	No	No
DHT22 (temperatura)	Compatible	No	No

Apartado energía:

Componentes del sistema:

Consumo total de los sensores y actuadores (considerando el consumo máximo para asegurar que no se subestime el requerimiento de energía):

DHT22: Consumo promedio diario: 180mWh/día.

Sensor PIR: Consumo promedio total diario: 31,45 mAh/día(aproximación).

Módulo RTC DS1307: consumo promedio total=180mWh/día.

Push button: 3.47mWh .

Pantalla LCD: consumo promedio diario de 3Wh/día.

Desagregado total del consumo de cada componente:	
Componente	**Consumo diario (mWh)**
DHT22 (Temperatura)	**180 mWh**
Sensor PIR	**31,45 mWh**
RTC DS1307	**180 mWh**
Push Button	**3.47mWh**
Pantalla LCD 20x4	**3 Wh**
Total del sistema diario	**3.395 Wh**

Consumo máximo, promedio y mínimo del sistema en mWh

El **consumo máximo** se refiere al caso en que todos los componentes están funcionando al máximo posible, el **consumo mínimo** sería si todos los componentes están en su estado de bajo consumo (reposo), y el **promedio** sería un valor intermedio.

Consumo máximo:

- **DHT22:** 180 mWh.
- **Sensor PIR:** 31.45 mWh (suponiendo que está activo continuamente).
- **RTC DS1307:** 180 mWh.
- **Push Button:** 3.47 mWh (considerando un uso diario).
- **Pantalla LCD 20x4:** 3 Wh (3,000 mWh).

Máximo total: aproximadamente 3.395 Wh.

Consumo mínimo:

- **RTC DS1307:** 180 mWh (funciona continuamente).
- **Push Button:** 0 mWh (no se presiona).
- **Pantalla LCD:** 0 mWh (apagada).
- **DHT22:** 0 mWh (sin realizar mediciones).
- **Sensor PIR:** 0.05 mA×5 V×24 h=6 mWh

· **Total mínimo diario:** 180+6=186 mWh.

Consumo promedio: Realizamos un promedio entre el mínimo y máximo consumo diario y obtenemos que el consumo promedio es de aproximadamente 1790,45 mWh.

Dimensionamiento de baterías para funcionamiento autónomo durante una semana:

- <u>**Consumo semanal del sistema:**</u>

  E semanal=E diario(Consumo maximo)×7.

E semanal=3.395Wh×7=23.765Wh.

Capacidad de la batería:

Con una batería de **5V**, la capacidad necesaria en mAh se calcula como:

Capacidad (mAh)=(E semanal / Voltaje de la batería)x1000

Capacidad (mAh)=23.765 Wh5 V×1000=4,753 mAh.

Selección de la batería:

Para garantizar una semana de autonomía necesitamos una batería con capacidad superior al consumo semanal calculado:

Capacidad mínima requerida: 4,753 mAh

Batería requerida: Batería de litio recargable (Li-ion) de **5V** y **5,000 mAh**.

Estrategia de Recarga:

Tiempo de recarga de la batería:

Tiempo de recarga (h)=Capacidad de la batería(mAh)/(Corriente de carga (mA) x eficiencia de carga).

Con una eficiencia de carga del 85% y una corriente de un volteo el tiempo de recarga resulta en:

Tiempo de recarga(h)=5000mAh/(1Ax0.85)=5.9 horas

Cargador USB: con un cargador de 5V y 1A y 85% de eficiencia la batería tendrá un tiempo estimado de recarga de **5 horas y 53 minutos**. Para evitar que el equipo quede inactivo durante este período, se puede optar por disponer de una **batería adicional** y alternar su uso según sea necesario.

Gabinete:

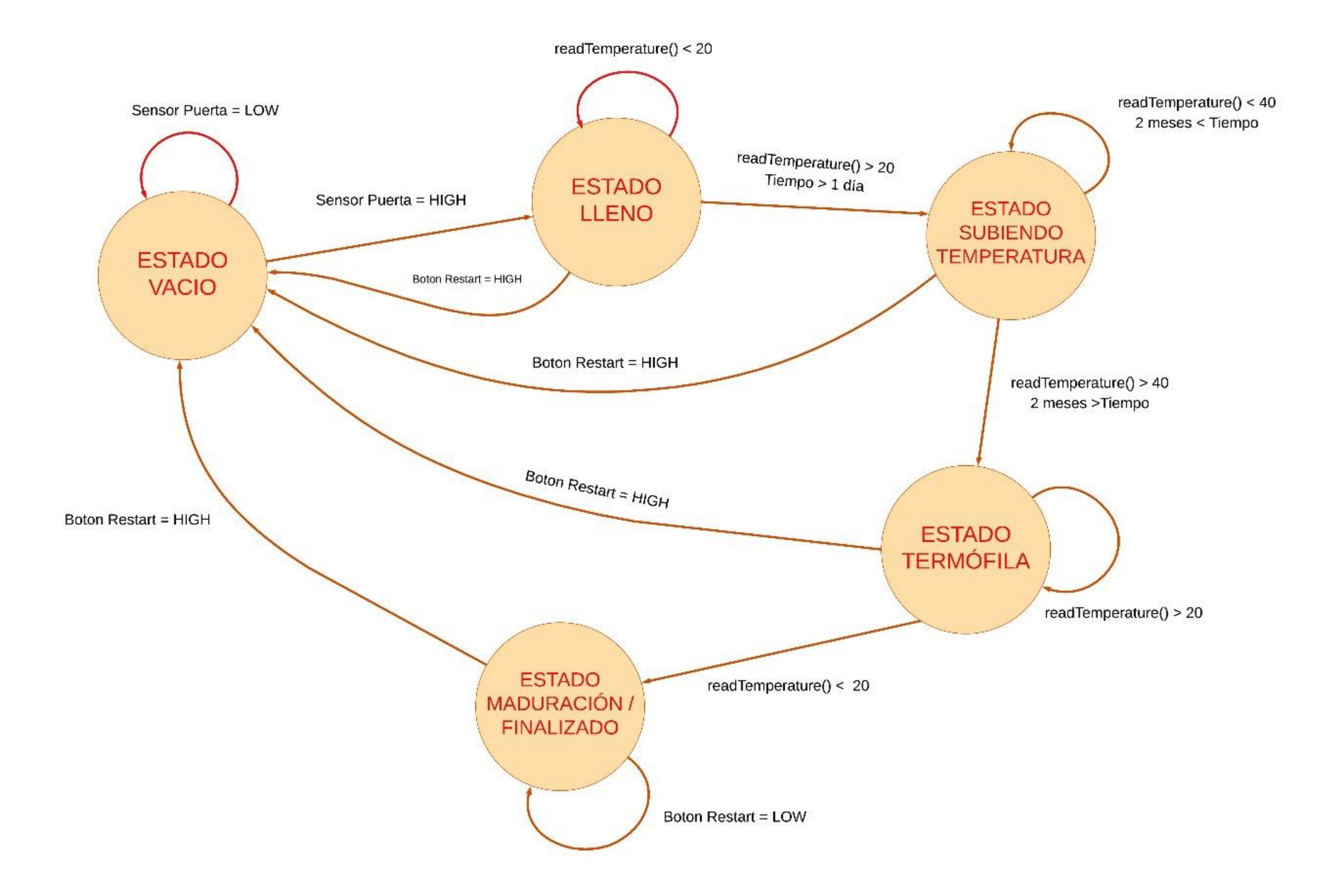
A la hora de elegir el compartimiento donde albergar todos los elementos necesarios debemos considerar varias cosas:

- **Material:** ya que trabajamos a la intemperie es necesario elegir un contenedor resistente tanto a la humedad de la lluvia como a la calidez del sol, además de al ser un proceso prolongado debe tener cierta durabilidad.
- **Protección:** No solamente debe ser a prueba de salpicaduras, sino también al calor generado por los componentes, es decir, tener una buena ventilación.
- **Accesibilidad:** Debe tener alguna forma de acceder a los componentes para mantenimiento o en caso de fallas.
- **Seguridad:** Al ser un ambiente donde no tenemos control las 24 horas del día, es prudente tomar ciertas medidas sobre el gabinete.

Teniendo en cuenta todo esto la mejor opción, y la más económica, es una caja de plástico reciclada (tupper), con agujeros en la parte inferior tapados con rejillas para mejorar el flujo del aire y evitar la entrada de insectos indeseados. La última consideración a tener en cuenta es agregar un agujero para pasar los cables necesarios y una vez puestos los mismos, sellarlos con silicona caliente para evitar la entrada del agua.

Si bien nombramos a la seguridad como factor relevante, concluimos que con la seguridad propia de la institución basta.

Diagrama de Estados:



Este diagrama refleja el flujo lógico del monitoreo continuo de temperatura con Arduino, destacando los estados y las transiciones entre ellos. A continuación, se describen, uno por uno, los distintos estados posibles:

1. **Inicio: Estado Vacío**

El sistema se inicializa, configurando los sensores y verificando que todo esté listo para comenzar. En este estado la compostera se encuentra vacía, esperando a que el sensor de movimiento (apertura de la compostera) produzca una entrada HIGH, dando por entendido el comienzo de carga de la misma. Así continuaremos al siguiente estado.

2. **Estado: lleno**

Estado posterior al estado vacío. Implica la carga completa de la compostera. En la primera etapa el sensor toma las lecturas de temperatura monitoreando si está por debajo de los 40 grados. En caso de superar esta cifra, pasa al siguiente estado.

3. **Estado: Aumento de temperatura**

Dentro de este estado, la temperaturas del compost comienzan a subir desde los 14°C a los 40°C, este estado tiene una duración aproximada de 2 a 4 meses. Pasado este tiempo, las temperaturas deberían oscilar entre los 35° C a 45° C. Si esto sucede, podríamos pasar al siguiente estado del compostaje.

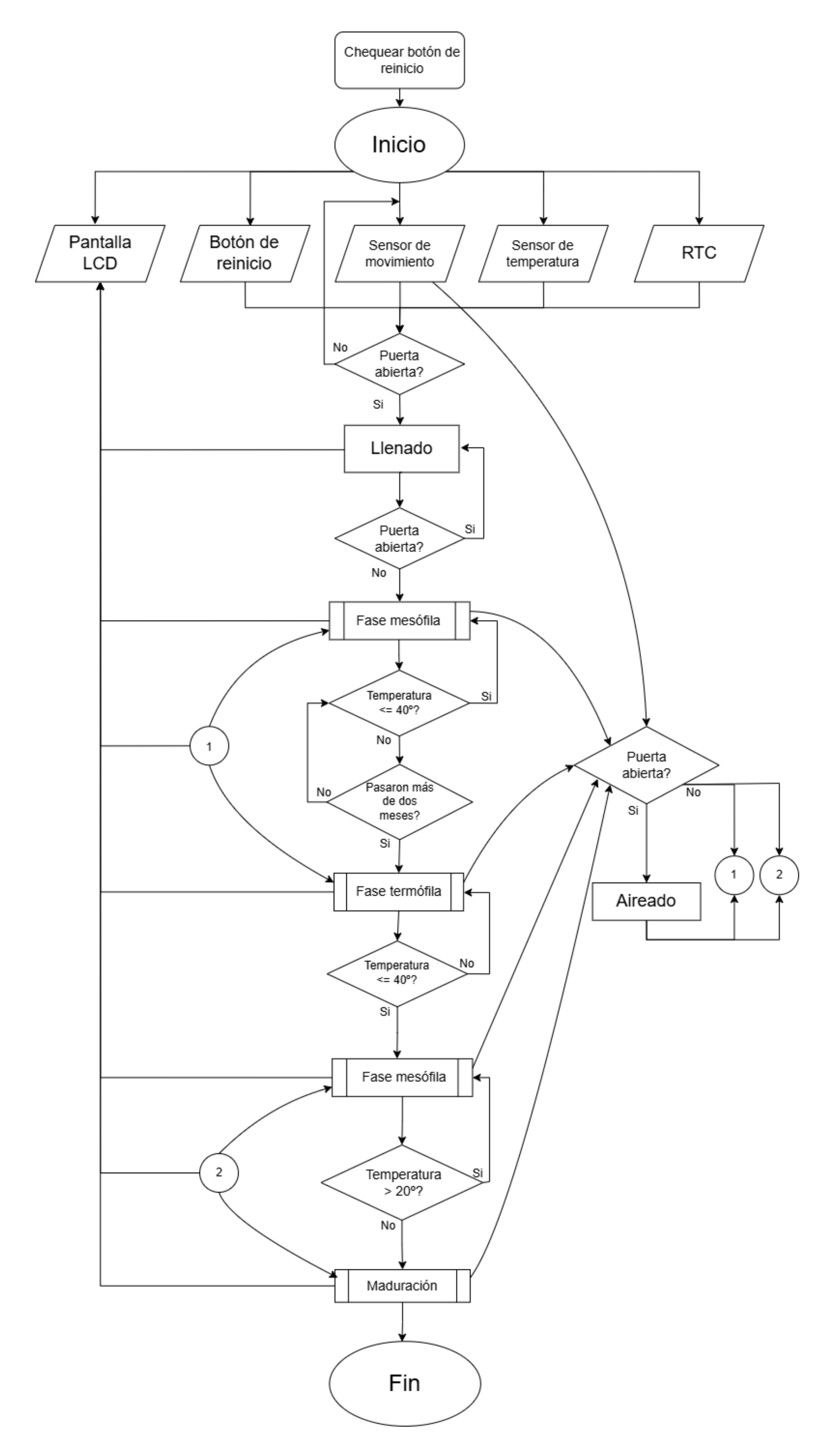
4. **Estado: Termófila.**

El pico más alto de temperatura donde oscilan los 40°C a 60°C (máximo rango de operación del sensor DHT22). Tras alcanzar estas temperaturas, la temperatura comenzará a decrecer, permitiendo avanzar al siguiente estado.

5. **Estado: Finalización/Maduración**

En este estado, el compost ha llegado a su etapa de maduración, listo para descargar y seguir su proceso para ser abono. El usuario encargado de esto, para poder comenzar nuevamente el ciclo del compostaje, deberá presionar un botón que enviará una entrada que nos devolverá al primer estado del Diagrama.

Diagrama de Flujo:



Transiciones clave del sistema:

1. **Vacío → Lleno:**

Ocurre tras recibir una entrada del sensor de movimiento (digitalRead(pirPin) == HIGH).

2. Lleno → Aumento de temperatura:

Se produce después de transcurrir 2 días y la temperatura sea mayor a 14°C.

3.Aumento de Temperatura → Termófila:

Se produce después de transcurrir un mínimo de 2 meses y la temperatura es mayor a 40°C.

4. Termófila → Maduración/ Finalización:

Se produce cuando la temperatura es menor a 20°C.

5.Maduración/ Finalización → Vacío:

Sucede cuando el usuario aprieta el botón de reinicio (digitalRead(buttonPin) == HIGH).

6. Lleno → Vacío:

Sucede cuando el usuario aprieta el botón de reinicio (digitalRead(buttonPin) == HIGH).

7. Aumento de temperatura → Vacío:

Sucede cuando el usuario aprieta el botón de reinicio (digitalRead(buttonPin) == HIGH).

8. Termófila → Vacío:

Sucede cuando el usuario aprieta el botón de reinicio (digitalRead(buttonPin) == HIGH).

Rangos de Temperatura:

Temperatura alta (>60°C): Riesgo de matar microorganismos beneficiosos.

Temperatura baja (<40°C): Actividad microbiana insuficiente.

Observaciones:

El estado de error puede añadirse para manejar fallas de sensores o problemas de conexión. El intervalo de espera debe ser configurable para ajustarse según la fase del compostaje. Opcionalmente, se puede incluir un registro de datos en una tarjeta SD o enviar las lecturas a un servidor remoto.

Código de Arduino:

```
// Librerías necesarias
#include <DHT.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include "RTClib.h"

// Definiciones de pines y tipo de sensor
#define DHTPIN 2       // Pin del botón
#define DHTTYPE DHT22 // Tipo de sensor de temperatura

// Inicialización del sensor DHT
DHT dht(DHTPIN, DHTTYPE);

// Variables globales
int contador_p = 0;          // Contador de aireaciones
DateTime startTime;          // Variable para guardar el tiempo inicial
const int pirPin = 4;        // Pin del sensor de movimiento
const int buttonPin = 5;     // Pin del botón de reinicio

// Definición de los estados del compostador
enum State {
  STATE_VACIO,
  STATE_LLENO,
  STATE_MESOFILA,
  STATE_TERMOFILA,
  STATE_FINALIZADO
};
```

```
State state = STATE_VACIO;

// Inicialización del display LCD
LiquidCrystal_I2C lcd(0x27, 20, 4);

// Inicialización del reloj RTC
RTC_DS1307 rtc;

// Función para mostrar "AIREANDO"
void aireando() {
  int i = 0;
  while (i != 3) {
    lcd.setCursor(0, 0);
    lcd.print("AIREANDO.          ");
    delay(1000);
    lcd.setCursor(0, 0);
    lcd.print("AIREANDO..          ");
    delay(1000);
    lcd.setCursor(0, 0);
    lcd.print("AIREANDO...          ");
    delay(1000);
    i++;
  }
  lcd.setCursor(0, 0);
  lcd.print("                    ");
}

// Función para mostrar "LLENANDO"
void llenando() {
  int i = 0;
  while (i != 3) {
    lcd.setCursor(0, 0);
    lcd.print("LLENANDO.          ");
    delay(1000);
    lcd.setCursor(0, 0);
    lcd.print("LLENANDO..          ");
```

```
    delay(1000);
    lcd.setCursor(0, 0);
    lcd.print("LLENANDO...         ");
    delay(1000);
    i++;
  }
  lcd.setCursor(0, 0);
  lcd.print("                    ");}


void setup() {
  Serial.begin(9600);

  // Inicializar el sensor DHT
  dht.begin();

  // Configuración de pines
  pinMode(pirPin, INPUT);      // Sensor de movimiento como entrada
  pinMode(buttonPin, INPUT);   // Botón de reinicio como entrada

  // Inicializar el display LCD
  lcd.init();
  lcd.backlight();
 // Inicializar el reloj RTC
  if (!rtc.begin()) {
    Serial.println("No se encontró el RTC");
    while (1);
  }

  if (!rtc.isrunning()) {
    rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
  }

  // Registrar el tiempo inicial
  startTime = rtc.now();
}
```

```
void loop() {
  float temperature = dht.readTemperature();  // Leer temperatura del
sensor
  DateTime currentTime = rtc.now();           // Obtener tiempo actual del
RTC

  // Mostrar temperatura y conteo en el display
  lcd.setCursor(0, 2);
  lcd.print("Temperatura: ");
  lcd.print(temperature, 1);
  lcd.print("C");

  lcd.setCursor(0, 3);
  lcd.print("Conteo aireado: ");
  lcd.print(contador_p);


  switch (state) {
    case STATE_VACIO:
      lcd.setCursor(0, 0);
      lcd.print("                    ");
      lcd.setCursor(0, 1);
      lcd.print("VACIO               ");

      if (digitalRead(pirPin) == HIGH) {
        llenando();
        startTime = currentTime;  // Guardar el tiempo de cambio de
estado
        state = STATE_LLENO;
      }
      break;

    case STATE_LLENO:
      lcd.setCursor(0, 0);
```

```
      lcd.print("                    ");
      lcd.setCursor(0, 1);
      lcd.print("LLENO               ");

      if (digitalRead(buttonPin) == HIGH) {
        state = STATE_VACIO;
      }

      if ((currentTime - startTime).totalseconds() >= 15 && temperature >
14) {
        startTime = currentTime;  // Actualizar tiempo para el siguiente
estado
        state = STATE_MESOFILA;
      }
      break;

    case STATE_MESOFILA:
      lcd.setCursor(0, 1);
      lcd.print("ELEVANDO TEMPERATURA");

      if (digitalRead(buttonPin) == HIGH) {
        state = STATE_VACIO;
      }

      if (digitalRead(pirPin) == HIGH) {
        contador_p++;
        aireando();
      }

      if ((currentTime - startTime).totalseconds() >= 15 && temperature >
40) {
        startTime = currentTime;
        state = STATE_TERMOFILA;
      }
      break;
```

```
    case STATE_TERMOFILA:
      lcd.setCursor(0, 1);
      lcd.print("MAX TEMPERATURA      ");

      if (digitalRead(buttonPin) == HIGH) {
        state = STATE_VACIO;
      }

      if (digitalRead(pirPin) == HIGH) {
        contador_p++;
        aireando();
      }

      if ((currentTime - startTime).totalseconds() >= 15 && temperature <
30) {
        startTime = currentTime;
        state = STATE_FINALIZADO;
      }
      break;

    case STATE_FINALIZADO:
      lcd.setCursor(0, 1);
      lcd.print("FINALIZADO          ");
      lcd.setCursor(0, 0);
      lcd.print("REINICIAR           ");
      contador_p = 0;

      if (digitalRead(buttonPin) == HIGH) {
        state = STATE_VACIO;
      }
      break;
  }
}
```

Desarrollo del proyecto:

Lo primero era entender la problemática y desglosarla en problemas más pequeños para manejarla con más facilidad. Lo más complicado de esto fue dividir las etapas del proceso de compost (fechas no son exactas) y el hecho de que no contamos con la capacidad de medir humedad directamente del compost sino que sólo del ambiente que lo rodea. Por esta razón, optamos por usar un estimado de dos meses como referencia y guiarnos únicamente por la temperatura, ignorando el nivel de humedad.

La realización de este proyecto requirió aplicar todo lo aprendido en la materia y más. Fue una excelente oportunidad para implementar todos los conceptos teóricos dados durante esta última mitad del año.

El área en la que más profundizamos fue en la sección de sensores y actuadores, clave a la hora de decidir cuáles usar y entender su funcionamiento. Además, el apartado de energía fue fundamental, ya que aprendimos sobre el consumo y su impacto en el diseño del sistema.

Una vez hecha la elección de componentes, el siguiente paso fue planificar el diseño del gabinete: decidir qué materiales utilizar, cómo evitar la entrada de agua de tal forma que el aire si tenga acceso, y sobre todo mantener los costos bajos. Con todo “armado”, el siguiente reto fue el apartado energético. Donde consideramos usar batería como fuente de alimentación, pero para elegir primero debíamos calcular el consumo total del sistema.

Finalmente, desarrollamos un gemelo digital para probar el funcionamiento de todo en conjunto y al quedar satisfechos con el resultado, documentamos todo por escrito siguiendo las indicaciones proporcionadas por los profesores de la cátedra.

Memoria técnica:

1) Justificación técnica:
   Para poder realizar eficientemente este proyecto, se necesita un sensor de temperatura, un sensor infrarrojo, un reloj, una pantalla LCD y un botón, esto con el fin de poder captar la temperatura, en qué tiempo la tuvo y transmitir esta información a través de la pantalla.

2) Proceso de ejecución:
   Una vez conseguidos todos los materiales del proyecto (sensores, actuadores, gabinete, placa Arduino) ya sea comprado por los proveedores que sugerimos o usando materiales usados, comience a conectar los componentes a la placa.

Una vez conectados, introducir en el gabinete los componentes sensibles a humedades o temperatura, también se debe de poner el código que les otorgamos al Arduino para que funcione correctamente. Hecho todo esto, solo quedará colocar todo en la compostera en la compostera.

REFERENCIAS:

Enlace de acceso al proyecto:
https://wokwi.com/projects/413733657687500801

Link Datasheets, compra y ejemplos de aplicación:

- **LCD Display:**

https://www.sparkfun.com/datasheets/LCD/HD44780.pdf

https://tienda.ityt.com.ar/display-lcd-caracter/2343-display-lcd-20x4-hd44780-i2c-azul-Arduino-itytarg.html

- **DS1307:**

https://www.alldatasheet.com/datasheet-pdf/view/123888/DALLAS/DS1307.html

https://www.todomicro.com.ar/temporizadores/365-modulo-rtc-ds1307-reloj-de-tiempo-real.html?srsltid=AfmBOor2LSHMDsyijVQEnGIMaIIulBKzP53Ma7hX8hD-Fwy32BRdZ22Z

https://wokwi.com/projects/303024107368219202

- **PIR MOTION:**

https://www.alldatasheet.es/datasheet-pdf/view/1131987/ETC2/HC-SR501.html

https://www.mercadolibre.com.ar/modulo-Arduino-sensor-de-movimiento-hc-sr501-pir-infrarrojo/p/MLA32490452#polycard_client=search-nordic&wid=MLA1665926544&sid=search&searchVariation=MLA32490452&position=1&search_layout=grid&type=product&tracking_id=368d6bd5-4e84-4e37-bb50-18ca96f7912d

- **DHT22:**

https://www.alldatasheet.com/datasheet-pdf/view/1132459/ETC2/DHT22.html

https://www.todomicro.com.ar/sensores/225-sensor-de-humedad-y-temperatura-dht22-Arduino.html?srsltid=AfmBOopMzkoCfZKxTzvaqg2sA1rY5HcPluKWCxd_OxzyWEqYKqX1jsS_

https://wokwi.com/projects/340367759893332564

- **Push Button:**

https://www.alldatasheet.com/datasheet-pdf/view/352476/CWIND/GPB502A05BR.html

https://articulo.mercadolibre.com.ar/MLA-708047127-10-x-push-button-boton-pulsador-tact-switch-6x6x8mm-Arduino-_JM

https://wokwi.com/projects/415006476837021697

- **DHT11:**

https://naylampmechatronics.com/sensores-temperatura-y-humedad/57-sensor-de-temperatura-y-humedad-relativa-dht11.html

- **DS18B20:**

https://tienda.bricogeek.com/sensores-temperatura/510-sensor-ds18b20-estanco.html

- **LM35:**

https://tienda.patagoniatec.com/productos/sensor-de-temperatura-lm35/

- **TCRT5000:**

https://sandorobotics.com.mx/producto/hr0214-123b/#:~:text=El%20TCRT5000%20es%20un%20sensor,objeto%20pasa%20enfrente%20del%20sensor.

- **GP2Y0A21YK:**

https://octopart.com/es/gp2y0a21yk-sharp-

40031513?gad_source=1&gclid=Cj0KCQiAi_G5BhDXARIsAN5SX7oKX9PkmGsp3eDmUPPydT5wxXCVuCCsdvW6TBEC_5CU1eJjm6pK1zkaAvxgEALw_wcB

- **ST7066U:**

https://es.aliexpress.com/i/1005005964738071.html

- **KS0066U:**

https://www.alldatasheet.es/datasheet-pdf/view/37317/SAMSUNG/KS0066U.html

Anexo fórmulas energía:

Consumo energético diario (en mWh o Wh):

$E = I \times V \times t$

Donde:

- E: Energía consumida (mWh o Wh).
- I: Corriente consumida (mA o A).
- V: Voltaje de operación (V).
- t: Tiempo de operación (horas).

Consumo energético semanal:

E semanal=E diario×7

Donde:

- E semanal: Energía consumida en una semana (Wh o mWh).
- E diario : Energía consumida en un día(Wh o mWh).
- 7: Días en una semana.

Capacidad necesaria de la batería:

Capacidad (mAh)=(E/V)×1000

Donde:

- Capacidad (mAh): Capacidad de la batería en miliamperios-hora.
- E: Energía total requerida (Wh).
- V: Voltaje de la batería (V).
- 1000: Conversión de Ah a mAh.

Consumo del botón por pulsación:

- Epulsación=$I \times V \times (t/3600)$

Donde:

- E pulsación : Energía consumida en una pulsación (mWh).
- I: Corriente consumida (mA).

- V: Voltaje de operación (V).
- t: Duración de la pulsación (segundos).
- 3600: Cantidad de segundos en una hora.

Consumo diario del botón (con N pulsaciones):

- E botón=Epulsación x N.

Donde:

- E botón : Energía consumida diariamente por el botón (mWh).
- E pulsación : Energía consumida en una pulsación (mWh).
- N: Número de pulsaciones diarias.

Tiempo de recarga de la batería:

- Tiempo de recarga (h)=Capacidad de la batería(mAh)/(Corriente de carga (mA) x eficiencia de carga)

Donde:

- Tiempo de recarga (h): Tiempo necesario para recargar completamente la batería.
- Capacidad de la bateria (mAh): Capacidad total de la batería en miliamperios-hora.
- Corriente de carga (mA): Corriente del cargador.
- Eficiencia de carga: porcentaje de la energía del cargador que llega efectivamente a la batería.

Consumo promedio diario del sistema:

- I promedio=(I reposo×t reposo+I activo ×t activo) / t total

  Donde:

- I promedio : Corriente promedio diaria (mA).
- I reposo : Corriente consumida en reposo (mA).
- I activo: Corriente consumida en estado activo (mA).
- t reposo : Tiempo en reposo (horas).
- t activo : Tiempo en estado activo (horas).

- t total : Tiempo total diario.